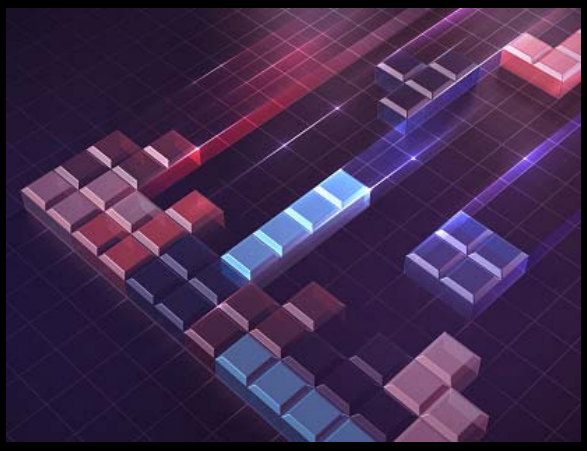
THREAD - ETAPE 8

Exécuter plusieurs objets dans une même JVM

BESOIN DE TRAITEMENTS EN PARALLÈLE

- des applications qui s'exécutent en parallèle au sein d'un même OS : multi-tasking

BESOIN DE TRAITEMENTS EN PARALLÈLE



- plusieurs objets graphiques évoluant simultanément dans une même interface

BESOIN DE TRAITEMENTS EN PARALLÈLE

- plusieurs objets actifs simultanément dans une même application
- plusieurs sous-processus au sein d'un processus => thread

THREAD JAVA

- Quand on lance un programme Java, on lance un processus (la JVM) contenant plusieurs *threads* :
  - Le *thread* principal (celui qui exécute le code à partir du ***main***),
  - Un *thread* pour la gestion du *garbage collector*,
  - …

THREAD JAVA

- Parallélisme : s'approcher de traitements simultanés

En réalité

- Un seul *thread* est actif simultanément sur un cœur du processeur,
- l'exécution des autres *threads* est alors suspendue : **parallélisme simulé.**

THREAD JAVA

- Contrairement au cas de différents processus qui ont chacun leur JVM, les thread d'un processus partagent le même espace mémoire.

⇒ un objet créé par un *thread* pourra être accessible par un autre *thread* (du même processus)

THREAD JAVA

- En JAVA un objet qui peut s'exécuter au sein d'un processus englobant est un objet « exécutable » ou *runnable*
- Le type ***Runnable*** est une *Interface*
  - => le traitement dépend complétement du problème métier à résoudre.
- Ce comportement est donné dans l'unique méthode *run* du type Runnable

CLASSE THREAD EN JAVA

- *Runnable* définit le comportement de l'objet exécutable
- le type *Thread* constitue le *"véhicule"* de l'objet exécutable :
  présente méthode *start()* qui démarre l'objet exécutable
- *Thead* implémente *Runnable* => avec une instance de *Thread* on a le comportement et le véhicule …

`java.lang.Thread`

THREAD JAVA

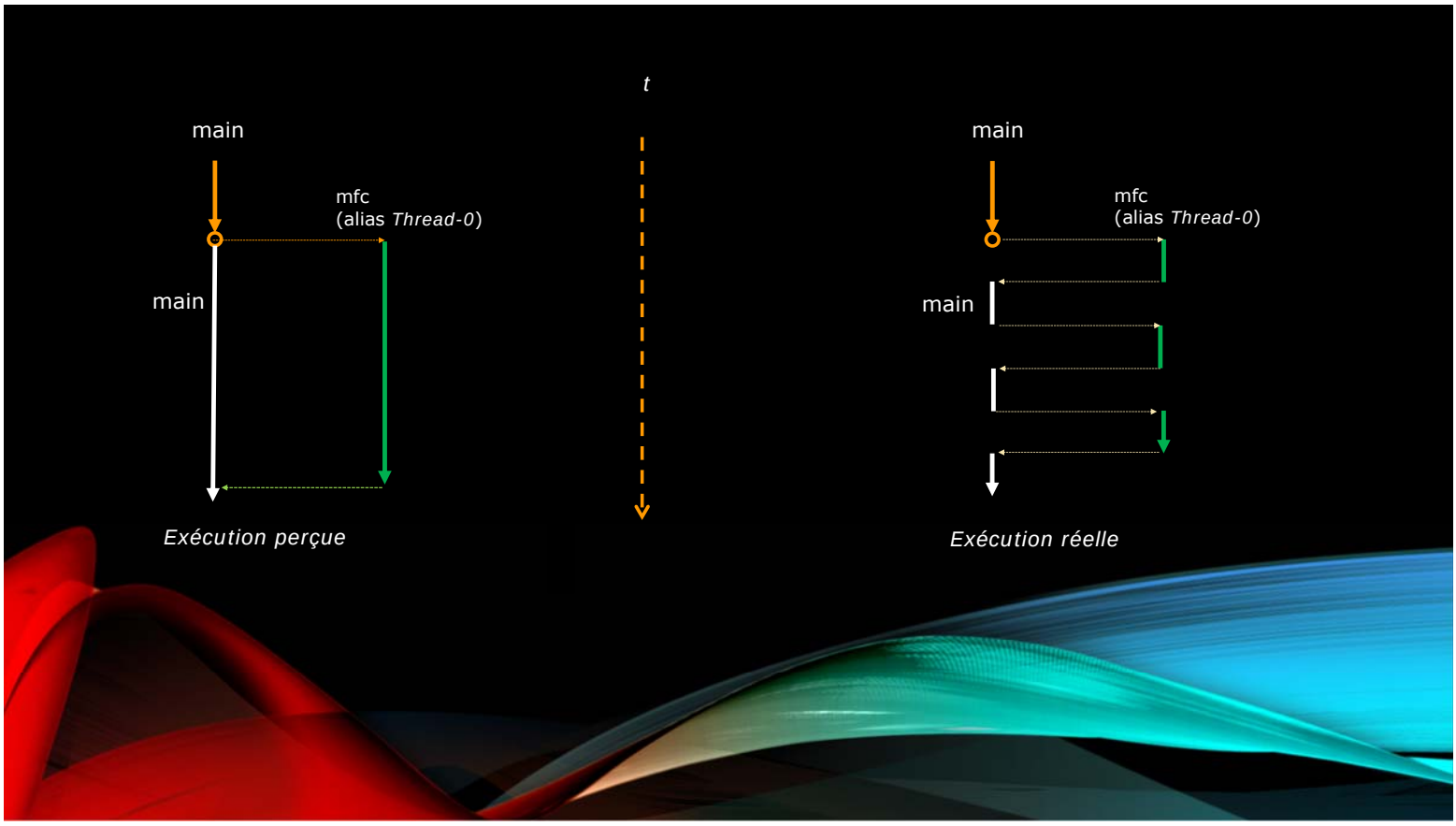
```
class SimpleCompteur extends Thread {

@Override
public void run() {
        Syso("je m'appelle " + this.getName() + " et je vais compter jusqu' à 10 ");

        for (int i = 1; i <= 10; i++) {
                        System.out.println("[" + this.getName() + "]:" +i);
                        }
        System.out.println("[" + this.getName() + ": J'ai fini !! ");
        }

}
```

THREAD JAVA

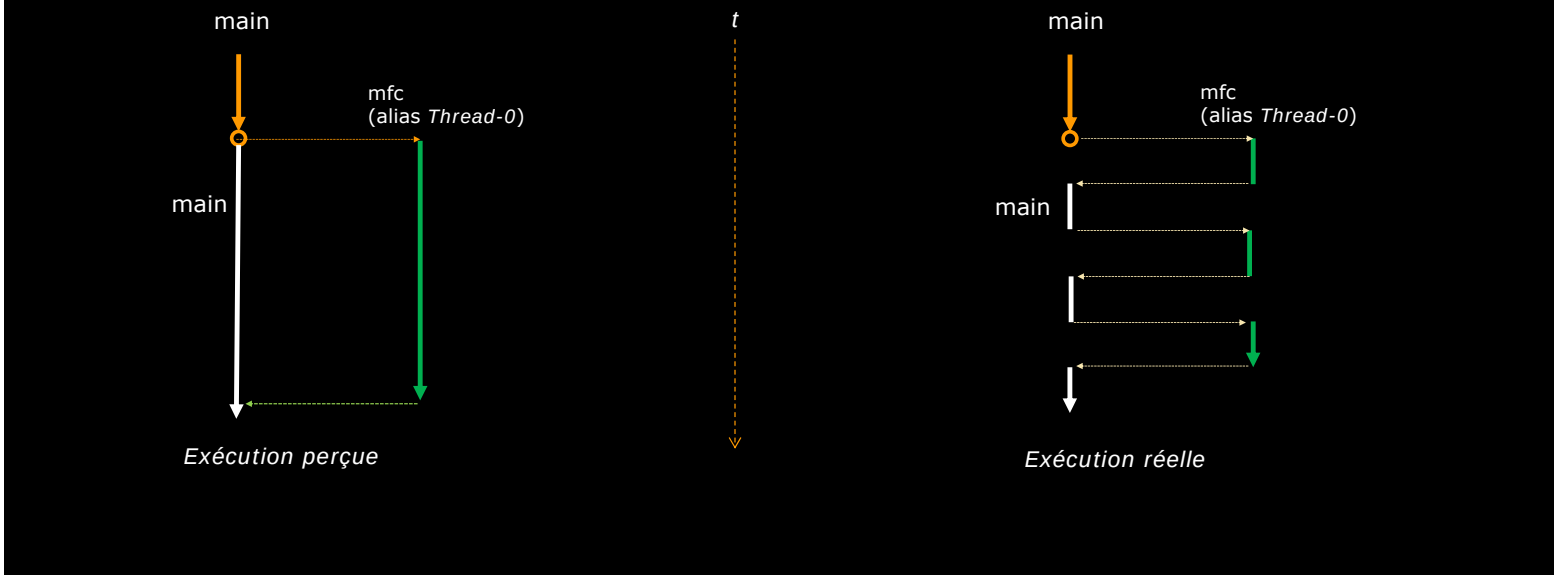
```java
public class Exemple1 {

public static void main(String[] args) {

    SimpleCompteur sc = new SimpleCompteur();

    sc.start();

    System.out.println("je suis le Main et je vais compter jusqu' à 10 ");
    for (int i = 1; i <= 10; i++) {
                            System.out.println("[Main]:" +i);
                            }
    System.out.println("[Main]: J'ai fini !! ");

    }
```

Extrait d'une trace d'exécution

```
je suis le Main et je vais compter jusqu' à 10
je m'appelle Thread-0 et je vais compter jusqu' à 10
[Main]:1
[Thread-0 ]:1
[Main]:2
[Thread-0 ]:2
[Thread-0 ]:3
[Thread-0 ]:4
[Thread-0 ]:5
[Thread-0 ]:6
[Main]:3
[Thread-0 ]:7
[Main]:4
[Thread-0 ]:8
[Main]:5
[Thread-0 ]:9
[Main]:6
[Thread-0 ]:10
[Main]:7
[Thread-0: J'ai fini !!
[Main]:8
[Main]:9
[Main]:10
```


t
main
mfc
(alias Thread-0)
main
Exécution perçue
main
mfc
(alias Thread-0)
main
Exécution réelle

THREAD JAVA


main
mfc
(alias Thread-0)
main
Exécution perçue
t
main
mfc
(alias Thread-0)
main
Exécution réelle